AS-0.1103 C-ohjelmoinnin peruskurssi
Aalto-yliopiston sähkötekniikan korkeakoulu
Tentti 25.10.2011, Raimo Nikkilä

Ohjeet

Kaikki ohjelmointitehtävät tulee toteuttaa C-kielellä hyvää ohjelmointityyliä noudattaen.

Tentti arvostellaan asteikolla 0-25p ja kaikkien tehtävien painoarvo on sama (5 pistettä / tehtävä).

Vastaa **viiteen** vapaavalintaiseen tehtävään.

Tentissä saa käyttää funktiolaskinta sekä tentissä jaettua C-kielen standardikirjaston minireferenssiä. Kaikki tentin tehtävät ovat tehtävissä minireferenssissä listatuilla funktioilla mutta kaikkia C-kielen standardikirjaston funktioita saa halutessaan käyttää.

Tavussa (`char`) on aina 8 bittiä kaikissa tehtävissä ja eniten merkitsevä bitti on aina vasemmalla.

Kirjoita edes opiskelijanumerosi selkeällä käsialalla tenttipapereihin.

1. Tehtävä

Vastaa lyhyesti seuraaviin C-kielen perusteita koskeviin kysymyksiin. Vastauksia ei tarvitse perustella mitenkään.

Vihje: Tentissä jaetusta minireferenssistä on apua erityisesti tässä tehtävässä

a) Oletetaan oheinen määritelmä:

```
1 char array[10];
```

Mitkä ovat nyt seuraavien lauseiden arvot?
`sizeof(array)`
`(&array[8] - &array[4])`

b) Toteuta oheinen `MIN`-makro siten että se evaluoituu pienemmäksi parametreistaan:
`#define MIN(L,R)`

c) Mikä tai mitkä oheisista funktiosta eivät voi huolehtia muistinvarauksesta tuottamalleen `int*` tyyppiselle arvolle:

(a) `void f(int *);`

(b) `void f(int **);`

(c) `int* f(void);`

(d) `int* f(int **);`

d) Määrittele `ptr` niminen funktio-osoitinmuuttuja, joka on tyypiltään sellainen, että siihen voitaisiin sijoittaa standardikirjaston funktio `realloc`.

e) Oletetaan oheiset määritelmät:

```
1 char *ptr;
2 ptr = malloc(20);
3 strcpy(ptr, "ABC");
```

Mitkä ovat nyt seuraavien lauseiden arvot?
`sizeof(ptr)`
`sizeof(*ptr)`
`strlen(ptr)`

2. Tehtävä

Toteuta funktio `RGBMix()` joka muodostaa annetuista RGB-väriarvoista sekoitevärin. RGB (Punainen, Vihreä, Sininen) väriavaruus on tapa esittää kaikki värit kolmen perusvärin kombinaationa.

```
1 unsigned char* RGBMix(const char*);
```

Funktio ottaa parametrinaan nimen tiedostolle josta sekoitettavat värit sekä niiden määrät luetaan. Lukeminen lopetetaan virheeseen tai kun tiedosto päättyy. Annetussa syötetiedostossa on rivejä muodossa:

```
Punainen Vihreä Sininen Määrä
```

Jossa värikomponentit ovat pieniä etumerkittömiä kokonaislukuja (`unsigned char`) ja määrä on liukuluku (`double`).

Funktio laskee sekoitevärin määrillä painotettuna keskiarvona ja palauttaa sen dynaamisesti (`malloc()`) varattuna kolmen alkion (`unsigned char`) taulukkona.

- Voit olettaa että `malloc()`-, `calloc()`- ja `realloc()`-funktiot onnistuvat aina.
- Mikäli syötetiedostoa ei saada avattua, funktio palauttaa `NULL`-arvon.
- Syötetiedoston lukemista jatketaan kunnes tiedosto loppuu tai lukemisessa tapahtuu virhe

Esimerkkisyötetiedosto:

```
128 0 0 0.5
0 128 0 1.0
0 128 128 2.5
```

Odotettu paluuarvo (taulukko): {16, 112, 80}
Arvot muodostuvat: 16 = (128 * 0.5 + 0 * 1.0 + 0 * 2.5) / 4.0 jne.

3. Tehtävä

Mitä alla oleva ohjelma tulostaa?

Vastaukseksi riittää pelkkä tuloste ilman perusteluja.

```
1  #include <stdio.h>
2
3  unsigned char function1(unsigned char m)
4  {
5    unsigned char c;
6    while (m)
7    {
8      c |= (m & 0x01);
9      m >>= 1;
10     c <<= 1;
11   }
12   return c;
13 }
14
15 size_t function2(unsigned char m1, unsigned char m2)
16 {
17   size_t n = 0;
18   for (size_t i = 0;i < 8;i++)
19     if (((m1 & m2) >> i) & 0x01)
20       n++;
21   return n;
22 }
23
24 int main()
25 {
26   unsigned char mask1 = 0xA3;
27   unsigned char mask2 = 0x6C;
28
29   printf("A: 0x%x 0x%x 0x%x\n", (unsigned char)~mask1, mask1 | mask2, mask1 & mask2);
30
31   printf("B: 0x%x 0x%x\n", mask2 << 1, mask1 >> 2);
32
33   printf("C: 0x%x 0x%x\n", mask1 + 5, mask2 - 3);
34
35   printf("D: 0x%x\n", function1(mask2));
36
37   printf("E: %zu %zu\n", function2(mask1, mask2), function2(mask2, mask1));
38 }
```

4. Tehtävä

Toteuta funktio `strsplit()`, joka hajottaa parametrina annetun merkkijonon korkeintaan kahteen osaan.

Funktio ottaa parametreinaan hajotettavan merkkijonon sekä merkin, jonka ensimmäisen esiintymän kohdalta merkkijono hajotetaan.

Funktio palauttaa hajotetun merkkijonon dynaamisesti (`malloc()`) varatussa merkkijonotaulukossa, jonka loppua ilmaisee `NULL` arvo.

```
1 char** strsplit(const char*, char);
```

- Voit olettaa että `malloc()`-, `calloc()`- ja `realloc()`-funktiot onnistuvat aina.
- Mikäli annettua merkkiä ei ole merkkijonossa, palautetaan taulukossa alkuperäinen merkkijono sekä tyhjä merkkijono.
- Annettua merkkiä ei sisällytetä kumpaankaan hajotetuista merkkijonoista.
- Mikäli annettu merkki esiintyy useamman kerran tai peräkkäin merkkijonossa, vain ensimmäinen ilmentymä huomioidaan.

Esimerkiksi syötteellä: `strsplit("huge-bunny", '-')`
Funktio tuottaa taulukon: {`"huge"`, `"bunny"`, `NULL`}

5. Tehtävä

Vastaa seuraaviin kysymyksiin alla olevan ohjelman perusteella.

a) Millaisella `typedef` määritelmällä voitaisiin yksinkertaistaa `fillArray()` funktion kolmatta parametria?

b) Millainen on funktion `dGenerator1()` oletuksena palauttama lukusarja?

c) Entä funktion `dGenerator2()`?

d) Anna esimerkki `main()` ohjelmasta joka luo ja täyttää `fillArray()` funktion avulla kuuden alkion taulukon {6.0, 12.0, 24.0, 48.0, 96.0, 192.0}

```
1  double dGenerator1(double init)
2  {
3    static double counter = 0;
4    if (init)
5      counter = init;
6    return counter++;
7  }
8
9  double dGenerator2(double init)
10 {
11   static double counter = 1;
12   if (init)
13     counter = init;
14   double temp = counter;
15   counter*=2;
16   return temp;
17 }
18
19 void fillArray(double *begin, double *end, double (*gen)(double))
20 {
21   while (begin != end)
22     *begin++ = gen(0);
23 }
```

6. Tehtävä

Esitetään kokonaisluvuista muodustavaa taulukkotietotyyppiä oheisella `intArray` tietueella. Tietueeseen kuuluu varsinainen kokonaislukutaulukko (`array`) sekä taulukon koko (`size`).

```
1 #include <stddef.h>
2
3 struct intArray
4 {
5   int* array;
6   size_t size;
7 };
```

Toteuta oheiselle tietotyypille seuraavat funktiot:

```
1 struct intArray newIntArray(size_t size);
2
3 void deleteIntArray(struct intArray);
4
5 void appendIntArray(struct intArray*, const struct intArray);
```

newIntArray() luo uuden `intArray` tietueen. Tietueen koko annetaan funktion parametrina ja kaikki taulukon alkiot alustetaan arvoon 0.

deleteIntArray() vapauttaa `intArray` tietueelle varatun muistin.

appendIntArray() lisää ensimmäisenä parametrina annetun `intArray` tietueen perään toisena parametrina annetun `intArray` tietueen sisällön.

- Voit olettaa että `malloc()`-, `calloc()`- ja `realloc()`-funktiot onnistuvat aina.
- 0 kokoisen `intArray` tietotyypin toiminta on määrittelemätön.

C99 minireferenssi

Tentissä tarvittavat tietotyypit ja niiden koot

Tyyppi	Koko tavuina	Lukualue	I/O muotoilumääre
`char`	1	-127 ... 127	%c
`unsigned char`	1	0 ... 255	%c
`int`	4	-2,147,483,647 ... 2,147,483,647	%d
`unsigned int`	4	0 ... 4,294,967,295	%u %x %X
`long`	8	Riittävän laaja ($-2^{63}...2^{63}-1$)	%ld
`size_t`	8	Riittävän laaja ($0...2^{64}-1$)	%zu
`double`	8	Riittävän laaja	%lf
`void*`	8	Kaikki (objekti-)osoitinmuuttujat	%p

Operaattoripresedenssi korkeimmasta matalimpaan ja assosioituvuus

`() [] . → ++ --`	vasemmalta oikealle	postfix ++ ja --
`++ -- + - ! ~ (tyyppi) * & sizeof`	oikealta vasemmalle	prefix ++ ja --, * ja & muistioperaattoreina
`* / %`	vasemmalta oikealle	* aritmeettisena operaattorina
`+ -`	vasemmalta oikealle	
`<< >>`	vasemmalta oikealle	
`< <= > >=`	vasemmalta oikealle	
`== != & ^ | && ||`	vasemmalta oikealle	& loogisena operaattorina
`?:`	oikealta vasemmalle	
`= op=`	oikealta vasemmalle	kaikki sijoitusoperaatiot
`,`	vasemmalta oikealle	pilkku operaattorina

C99 varatut sanat

`auto`	`break`	`case`	`char`	`const`	`continue`	`default`	`do`
`double`	`else`	`enum`	`extern`	`float`	`for`	`goto`	`if`
`inline`	`int`	`long`	`register`	`restrict`	`return`	`short`	`signed`
`sizeof`	`static`	`struct`	`switch`	`typedef`	`union`	`unsigned`	`void`
`volatile`	`while`	`_Bool`	`_Complex`	`_Imaginary`			

Tentissä tarvittavat standardikirjaston funktiot

ctype.h

```
int isalnum(int ch);   int isalpha(int ch);   int isdigit(int ch);   int islower(int ch);
int ispunct(int ch);   int isspace(int ch);   int isupper(int ch);   int tolower(int ch);
int toupper(int ch);
```

math.h

```
double pow(double base, double exponent);   double sqrt(double value);
```

stdio.h

```
int printf(const char* format, ...);
int sprintf(char* str, const char* format, ...);
int fscanf(FILE* stream, const char* format, ...);
FILE* fopen(const char* path, const char* mode);
int getchar(void);
int putchar(int ch);
int puts(const char* str);
char* fgets(char* str, int size, FILE* stream);
int fprintf(FILE *stream, const char* format, ...);
int scanf(const char* format, ...);
int sscanf(const char* str, const char* format, ...);
int fclose(FILE* fp);
int fgetc(FILE *stream);
int fputc(int ch, FILE* stream);
int fputs(const char* str, FILE* stream);
```

stdlib.h

```
void* calloc(size_t nmemb, size_t size);   void free(void* ptr);   void abort(void);   int abs(int);
void* realloc(void* ptr, size_t size);   void* malloc(size_t size);   void exit(int);
void qsort(void* base, size_t nmemb, size_t size, int (*cmp)(const void*, const void*));
```

string.h

```
char* strcat(char* dest, const char* src);
int strcmp(const char* str1, const char* str2);
char *strstr(const char* haystack, const char* needle);
size_t strxfrm(char* dest, const char* src, size_t n);
char* strchr(const char* str, int ch);
char* strcpy(char* dest, const char* src);
size_t strlen(const char *str);
void* memset(void* s, int c, size_t n);
```